

MVC - Controller and View

Charles Severance

Textbook: Build Your own Ruby on Rails Application by Patrick Lenz (ISBN:978-0-975-8419-5-2)

The Rails Approach...

- In Rails, everything has its place and everything is in its place.
- Initially this seems like a pain - but it is wonderful when looking at many different Rails applications.
- Rails effectively pre-chooses 90% of the architecture choices of a web application.
- We learn and follow the Rails approach

Ruby on Rails vs PHP - RailsEnvy.com Commercial #3

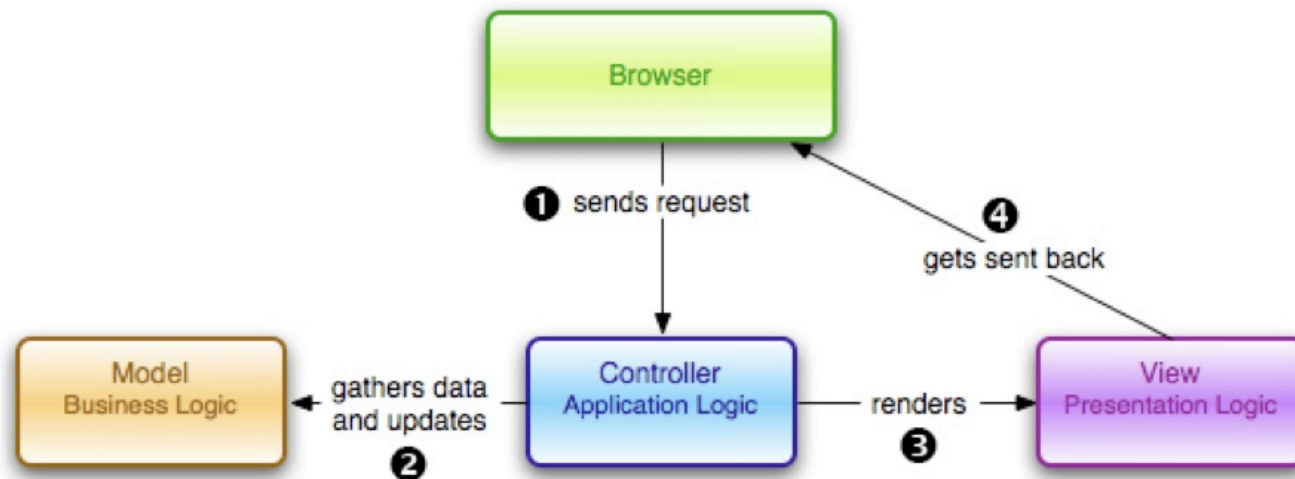


<http://www.youtube.com/watch?v=p5ElrSM8dCA>

Model - View - Controller

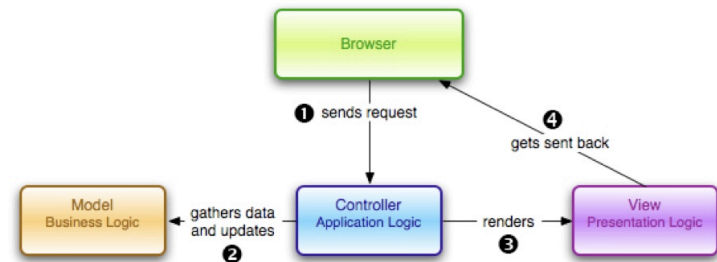
- Model - Persistent Storage - Database - stores across all sessions and across time
- View - Look and Feel of the application - usability, accessibility, design, appeal, functionality, drag and drop,
- Controller - Program logic, business rules, flow from screen to screen controls each session independently.

MVC - Request - Response Cycle



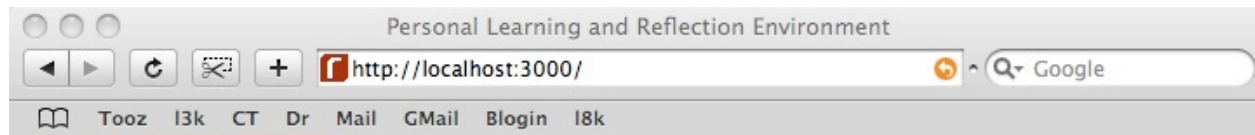
MVC Sequence

- User presses button, browser sends data to application
- Controller receives the data, and makes updates to and/or retrieves from the model as necessary
- User output data is passed to the View - view applies final look and feel and the response goes back to the Browser.



MVC Detail

- An application can have many controllers
- Each controller can have a number of actions - actions roughly correspond to the buttons in an application - what you want to be done
- Each controller can have many views - views roughly correspond to the screens in an application
- Each application can have many models to store data - and there are relationships between those models



Toozday Personal Learning Environment (11.0)

You are logged in as admin ([logout](#))

Three Controllers

[Refresh \(0 Sites\)](#) [My Profile](#) [Users](#) [Sites](#)

Add Account

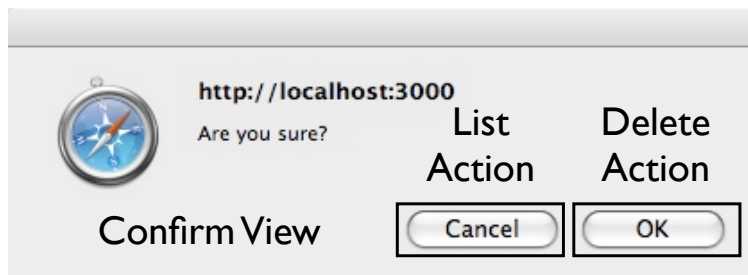
Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete

The portal shows two tools at a time using Ajax and they operate independently. Site tools work within the context of that site.

///

Add Account

Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete



Views: List, Add, View, Confirm

Actions: List, Add, Create, View,
Confirm, Delete

Please in Fields for your New Account

Name:

Login:

Password:

Email:

Create

Cancel

Show All

To Edit, simply click on the text you want to edit

Name:

Trek Glowaki

[edit](#)

Login:

trek

[edit](#)

Password:

trek

[edit](#)

E-Mail:

[edit](#)

You are logged in as admin ([logout](#))

[Add Account](#)

[My Profile](#) [Users](#) [Sites](#) [Fun](#)

Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete

GET /users/list

1

Model(s)

Profile

Belongs to

Belongs to

Sites

chuck
trek
beth

2

List

View

Add

Create

Confirm

Delete

Controller

3

4

List

Detail

Add

Confirm

Views

Add Account

Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete

Please in Fields for your New Account

Name:

Login:

Password:

Email:

Create | Cancel

GET /users/add

Model(s)

Profile

Belongs to

Belongs to

Sites

User

List

View

Add

Create

Confirm

Delete

Controller

List

Detail

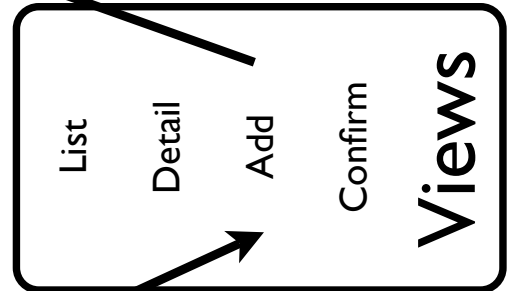
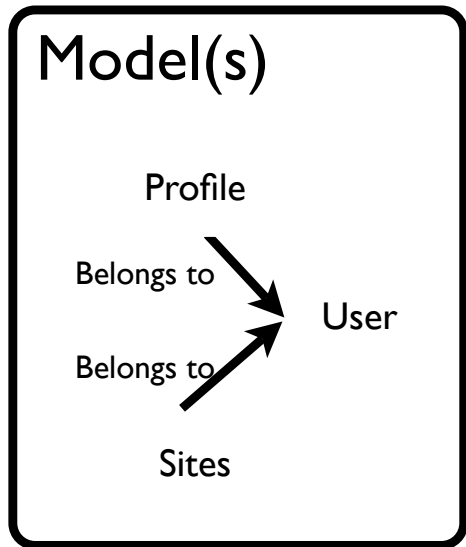
Add

Confirm

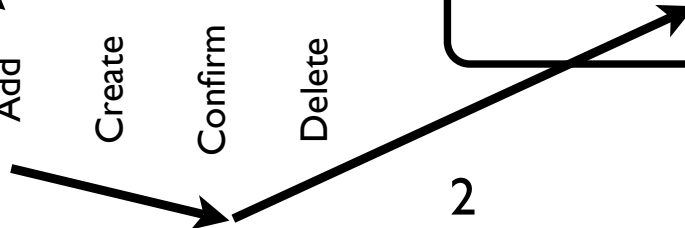
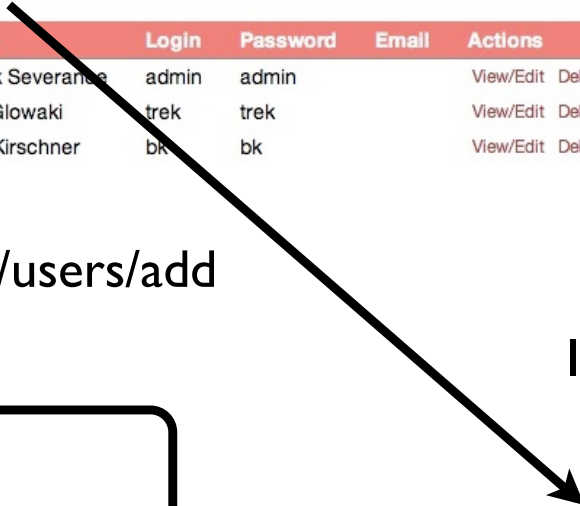
Views

3

2



GET /users/add



Please in Fields for your New Account

Name:

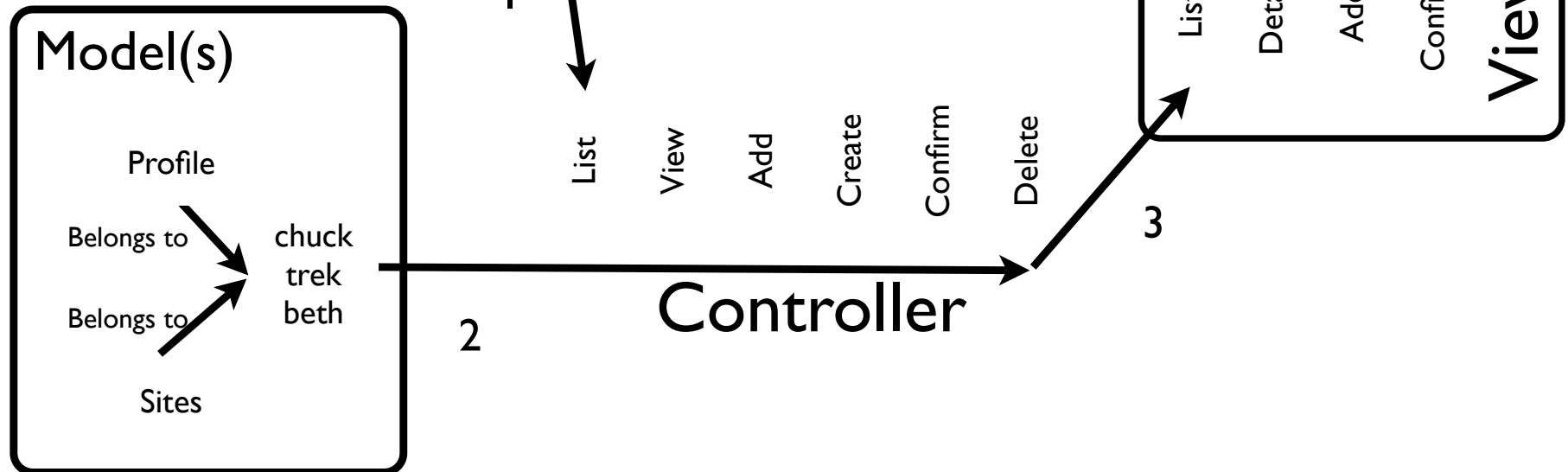
Login:

Password:

Email:

Add Account

Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete



Please in Fields for your New Account

Name:

Janet Chen

Login:

janet

Password:

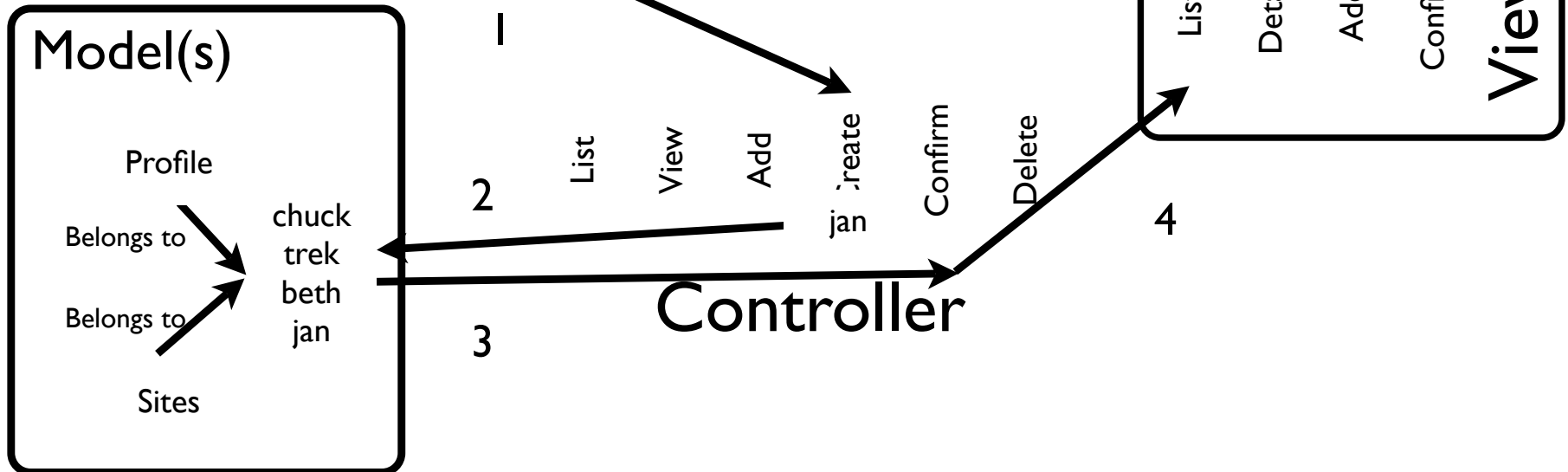
Email:

j@an.net

Create Cancel

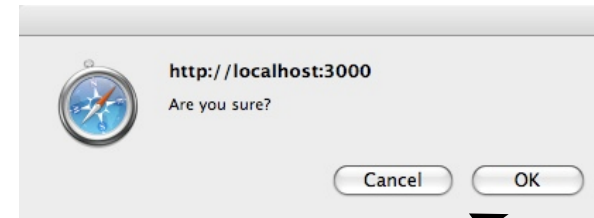
Add Account

Name	Login	Password	Email	Actions
Charles Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete
Janet Chen	janet	janet	j@an.net	View/Edit Delete

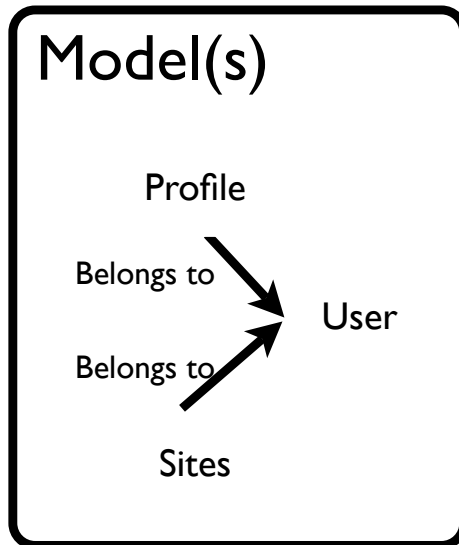


Add Account

Name	Login	Password	Email	Actions
Charles Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete
Janet Chen	janet	janet	j@an.net	View/Edit Delete

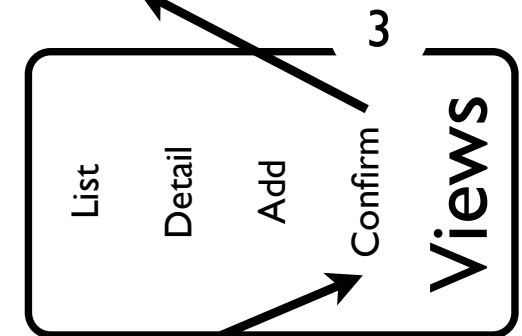


GET /users/confirm



List
View
Add
Create
Confirm
Delete

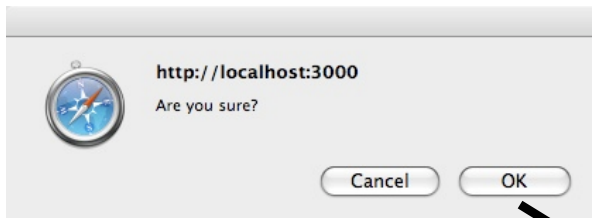
Controller



2

3

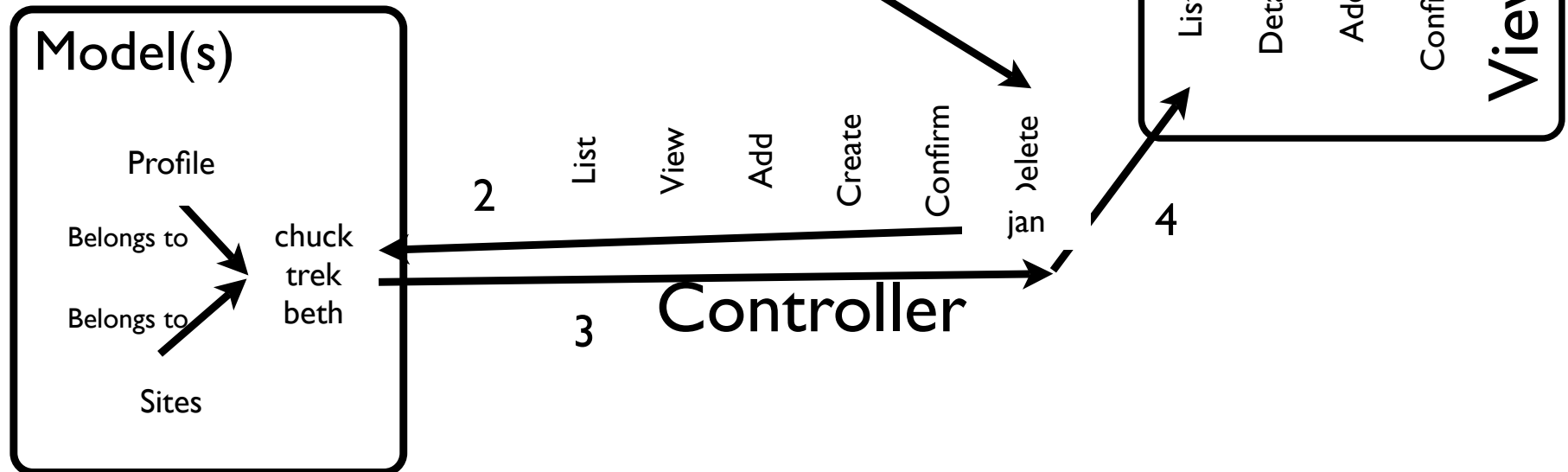
1



Add Account

Name	Login	Password	Email	Actions
Chuck Severance	admin	admin		View/Edit Delete
Trek Glowaki	trek	trek		View/Edit Delete
Beth Kirschner	bk	bk		View/Edit Delete

POST /users/delete

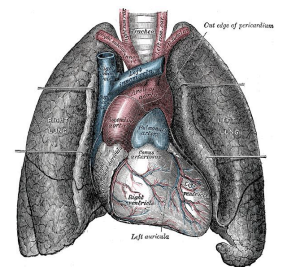


Controllers, Actions, Views

- Each controller has one or more actions
- Actions correspond to “asking for something to be done”
- Views correspond to the pages you see

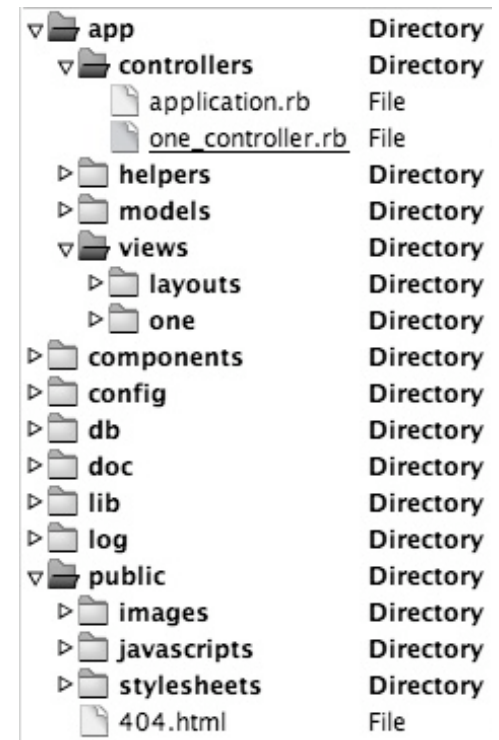
Rails Controller Anatomy

<http://en.wikipedia.org/wiki/Image:Heart-and-lungs.jpg>



Inside A Rails Application

- Rails dictates the layout of an application directory - one less decision for a developer to make
- The directory structure precisely reflects the MVC architecture
- This helps Rails developers know where to look for things when faced with a new Rails application
- Removing choice improves clarity and speeds learning

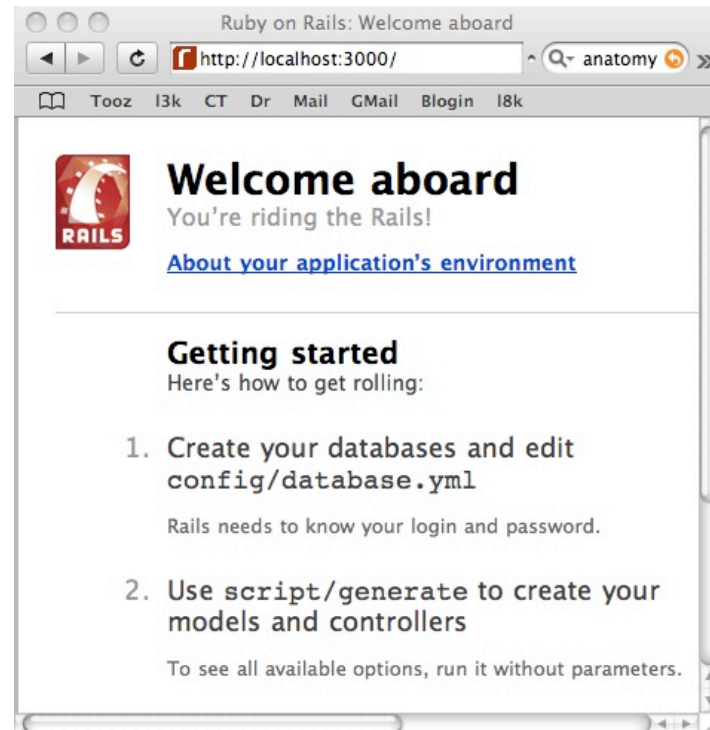
A screenshot of a file explorer showing the directory structure of a Rails application. The structure is organized into several main directories: 'app' (containing 'controllers', 'helpers', 'models', 'views', 'components', 'config', 'db', 'doc', 'lib', 'log', and 'public'), 'public' (containing 'images', 'javascripts', and 'stylesheets'), and a '404.html' file. The 'app' directory is expanded, showing its sub-directories and files. The 'controllers' directory contains 'application.rb' and 'one_controller.rb'. The 'views' directory contains 'layouts' and 'one'. The 'public' directory contains 'images', 'javascripts', and 'stylesheets'. The '404.html' file is located at the root of the application directory.

▼ app	Directory
▼ controllers	Directory
application.rb	File
one_controller.rb	File
▶ helpers	Directory
▶ models	Directory
▼ views	Directory
▶ layouts	Directory
▶ one	Directory
▶ components	Directory
▶ config	Directory
▶ db	Directory
▶ doc	Directory
▶ lib	Directory
▶ log	Directory
▼ public	Directory
▶ images	Directory
▶ javascripts	Directory
▶ stylesheets	Directory
404.html	File

Making a Rails Application

- The rails command makes a new folder / directory and populates it with a skeleton Rails application.
- The application is fully functional and had a lot of structure but it has no models, views or controllers.
- Rails puts in all the places where models, views, and controllers go - but they are empty.

```
$ rails app0 -d sqlite3
  create app/controllers
  create app/controllers/application.rb
  create app/helpers
  create app/helpers/application_helper.rb
  create app/models
  create app/views/layouts
  create db
  create public/index.html
  create public/images/rails.png
  create public/stylesheets
  create public/javascripts/prototype.js
  create public/javascripts/effects.js
  create log
  create log/server.log
  create log/production.log
  create log/development.log
  create log/test.log
$ cd app0
$ ruby script/server
```



Controller, Actions, Views






- We use a code generator to add a controller with some views to our application

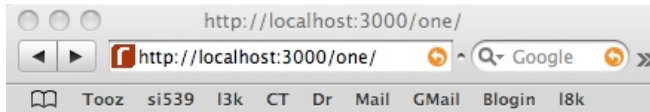
```
$ cd app0
```

```
$ ruby script/generate controller One index add
```

Please generate the code for a controller named “One” with actions named “index” and “add” and views named “index” and “add”.

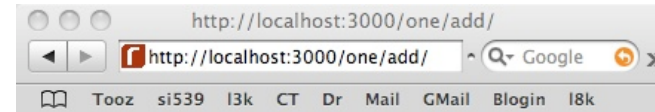
```
$ ruby script/generate controller One index add
  exists app/controllers/
  exists app/helpers/
  create app/views/one
  exists test/functional/
  create app/controllers/one_controller.rb
  create test/functional/one_controller_test.rb
  create app/helpers/one_helper.rb
  create app/views/one/index.rhtml
  create app/views/one/add.rhtml
$ ruby script/server
=> Booting ....
```

▼  app	Directory
▼  controllers	Directory
 application.rb	File
 one_controller.rb	File
▶  helpers	Directory
▼  models	Directory
▼  views	Directory
▼  layouts	Directory
▼  one	Directory
 add.rhtml	File
 index.rhtml	File
-	



One#index

Find me in app/views/one/index.rhtml



One#add

Find me in app/views/one/add.rhtml

//

//

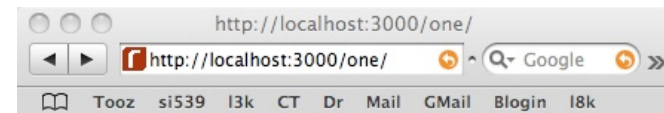
http://localhost:3000/one/add/

Controller
Action

The default action is assumed to be “index”.

A View

- Rails has generated a simple view template for us because we requested an index view
- Rails even put a hint as to where to find the view file.
- This view ends in rhtml - this means that it is not just HTML, it also can contain Embedded Ruby code.



One#index

Find me in app/views/one/index.rhtml

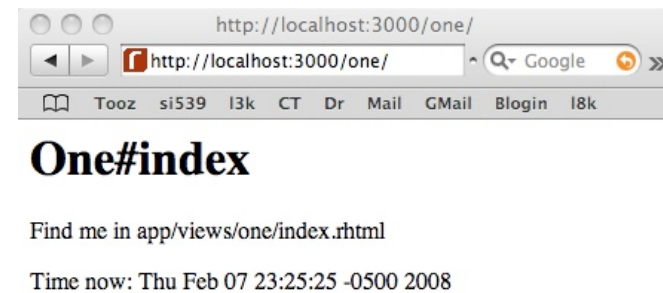


Embedded Ruby

- In rhtml files, text between `<%` and `%>` is ruby code which is executed
- When something is between `<%=` and `%>` it is printed as part of the returned HTML



```
index.rhtml (/Users/csevadadmin/rails_apps/app0/app/views/one/)
<h1>One#index</h1>
<p>Find me in app/views/one/index.rhtml</p>
<p>Time now: <%= Time.now %></p>
```



Adding an Image

- We store images under the directory public/images
- We use the image_tag helper in Embedded Ruby to generate the HTML img tag.
- We tell image_tag the name of the image file.

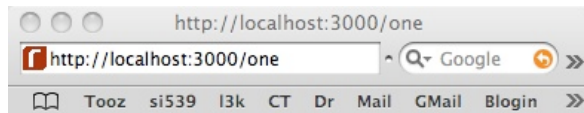


<p><%= image_tag "rails.png", :alt => "Rails Logo" %></p>

<p>Find me in app/views/one/index.rhtml</p>

<p>Time now: <%= Time.now %></p>

<p><%= image_tag "rails.png", :alt => "Rails Logo" %></p>



Find me in app/views/one/index.rhtml

Time now: Fri Feb 08 00:13:10 -0500 2008



Rails adds a timestamp to the end of the image reference to make sure the browser does not cache images and possibly ignore a change when you update an image.

<p>Find me in app/views/one/index.rhtml</p>

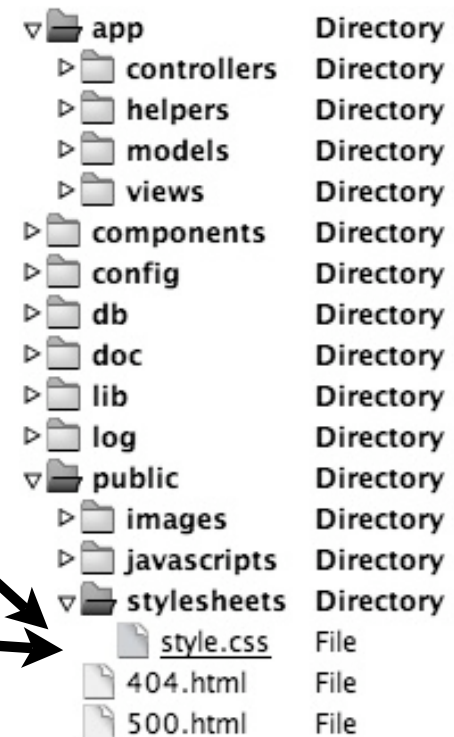
<p>Time now: Fri Feb 08 00:13:10 -0500 2008</p>

<p></p>

Cascading StyleSheets

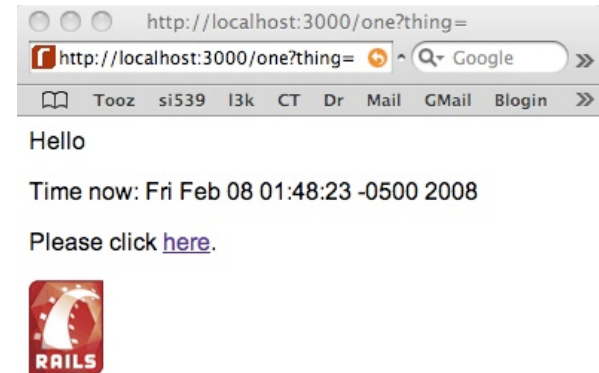
- Style sheets are stored under public/stylesheets
- There is an embedded Ruby helper to reference a stylesheet

`<%= stylesheet_link_tag "style.css" %>`



```
body {  
  font-family: arial;  
}
```

```
<html>  
<head>  
  <%= stylesheet_link_tag "style.css" %>  
</head>  
<body>  
<p><%= @greet %></p>  
<p>Time now: <%= Time.now %></p>  
<p>Please click  
<a href="<%= url_for :action => "add" %>" >here</a>.  
<p><%= image_tag "rails.png", :alt => "Logo" %></p>  
</body>
```



Up Next: Big Picture

Linking Between Actions

- Even though we know the urls for an action we have a special utility in Rails that gives links to actions within controllers.
- The `url_for` - generates a url which references an action

`url_for :action => "add"`

In English - Make me a URL which will get me to the add action.

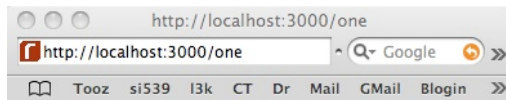
<p>Find me in app/views/one/index.rhtml</p>

<p>Time now: <%= Time.now %></p>

<p>Please click

<a href="<%= url_for :action => "add" %>" >here.

<p><%= image_tag "rails.png", :alt => "Rails Logo" %></p>



Find me in app/views/one/index.rhtml

Time now: Fri Feb 08 00:20:06 -0500 2008

Please click [here](#).



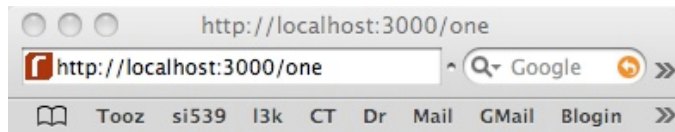
<p>Find me in app/views/one/index.rhtml</p>

<p>Time now: Thu Feb 07 23:39:19 -0500 2008</p>

<p>Please click

here.

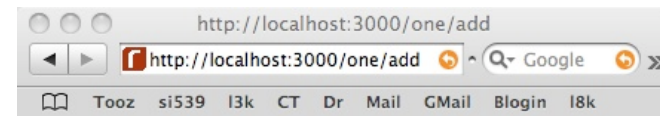
<p></p>



Find me in app/views/one/index.rhtml

Time now: Fri Feb 08 00:20:06 -0500 2008

Please click [here](#).



One#add

Find me in app/views/one/add.rhtml

```
<h1>One#index</h1>
```

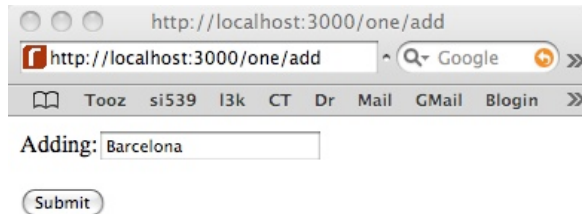
```
<p>Find me in app/views/one/index.rhtml</p>
```

```
<p>Time now: <%= Time.now %></p>
```

```
<p>Please click
```

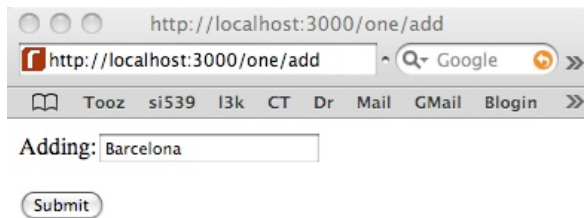
```
<a href="<%= url_for :action => "add" %>" >here</a>.
```

```
<form action="<%= url_for :action => "index" %>" method="get">
  <p>Adding:<input type="text" name="thing"/> </p>
  <p><input type="submit" /></p>
</form>
```

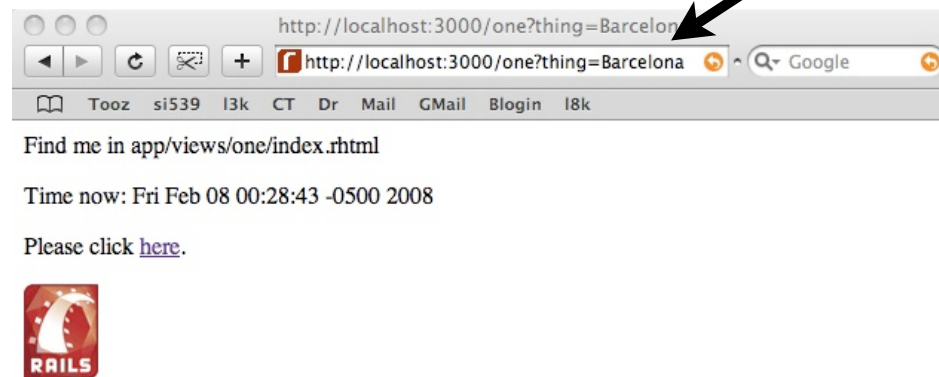


We make a simple form and use embedded Ruby and the `url_for` command to generate a url to use for form submission. We will submit our form input to the index action.

```
<form action="/one/index" method="get">
  <p>Adding:<input type="text" name="thing"/> </p>
  <p><input type="submit" /></p>
</form>
```

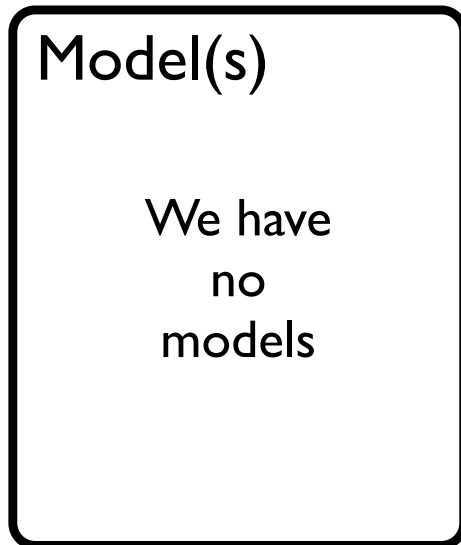
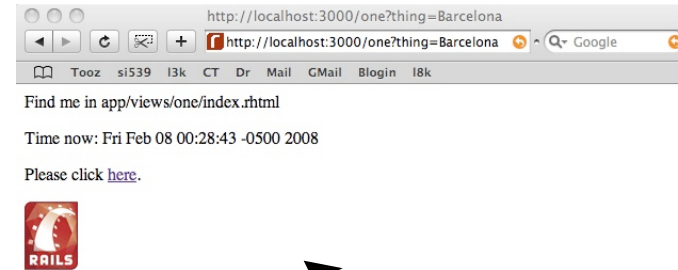
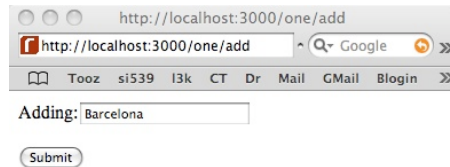
```
<form action="/one/index" method="get">  
  <p>Adding:<input type="text" name="thing"/> </p>  
  <p><input type="submit" /></p>  
</form>
```



With the GET method, the form data is appended to the URL after a question mark (?).

Inside the Action

- Even though we have been using actions, we really have been focused on learning on how to work in the view.
- Effectively we have been using empty or “do nothing” actions.



Index
Add

We have 2 actions.

One

We have 2 views.



Our actions don't do anything they just drop through to the views.

1

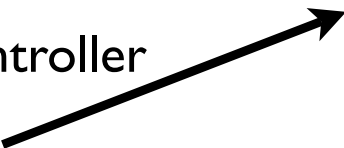
3

2

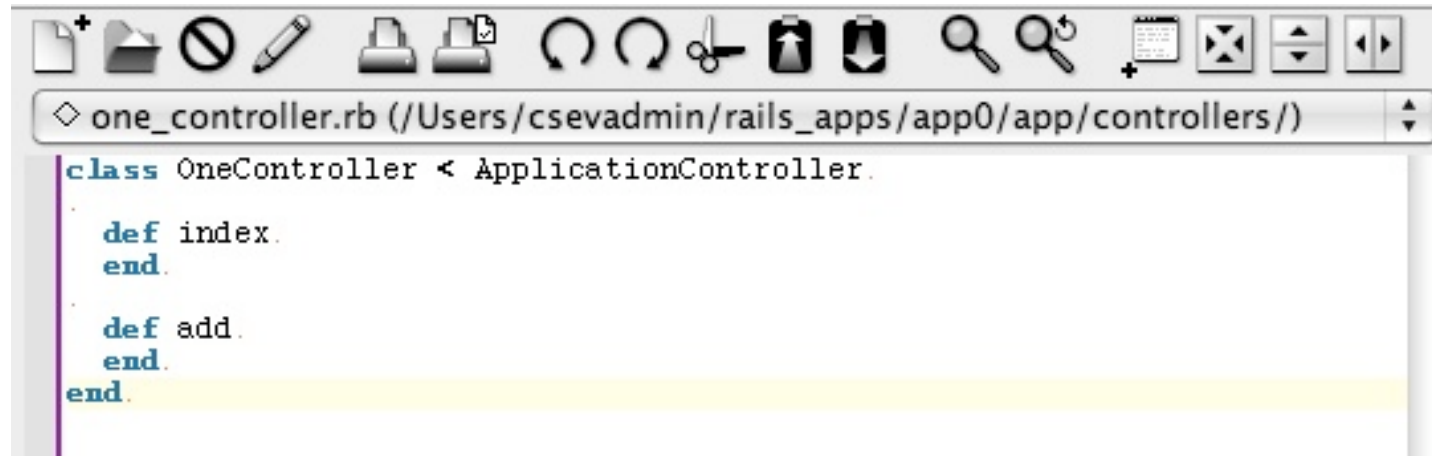
It is Time for Action!

Where is the Action at?

- The actions for a particular controller are all stored in a file called `one_controller.rb`
- The controller is all written in the Ruby programming language



▼ app	Directory
▼ controllers	Directory
application.rb	File
one_controller.rb	File
helpers	Directory
models	Directory
views	Directory
layouts	Directory
one	Directory
add.rhtml	File
index.rhtml	File
components	Directory
config	Directory
db	Directory
doc	Directory
lib	Directory
log	Directory
public	Directory



```
◇ one_controller.rb (/Users/csevaadmin/rails_apps/app0/app/controllers/)
class OneController < ApplicationController.
  def index.
  end.

  def add.
  end.
end.
```

- This file has two actions - one named “index” and the other named “add” - just like we requested on the ruby script/generate controller command many slides back.
- Both actions are doing nothing at this point in time and just fall through to the view of the same name. The “add” action falls through into the add.rhtml view file.

```
class OneController < ApplicationController
  def index
  end

  def add
  end
end
```

What a controller looks like. At least what a controller with actions that do nothing except fall through to the view files looks like.

Index

Add

We have 2 actions.

One

Lets get into Action!

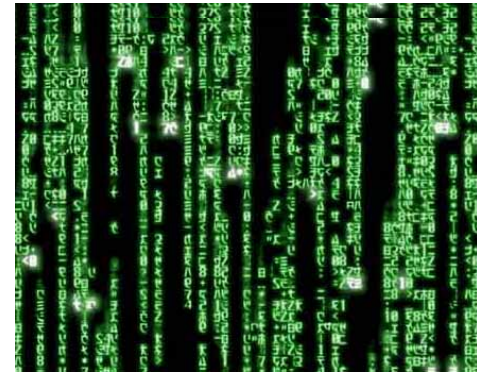
- A simple thing to do is to add the Rails code to print a message in the log whenever the action is executes.

```
class OneController < ApplicationController
  def index
    logger.info "WE ARE IN THE INDEX ACTION!"
  end

  def add
    logger.info "WELCOME TO THE ADD ACTION"
  end
end
```

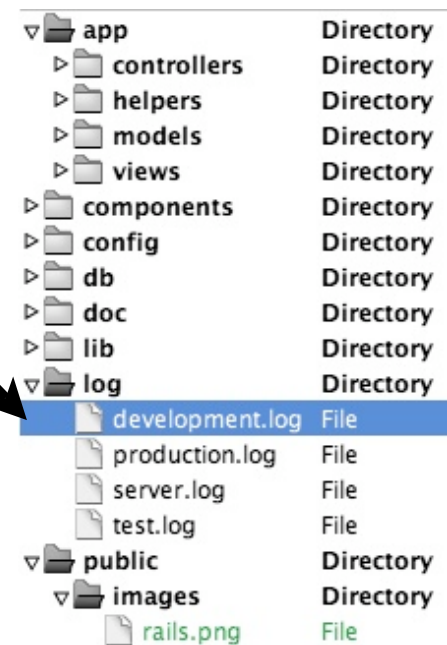

Watching the Rails Logs

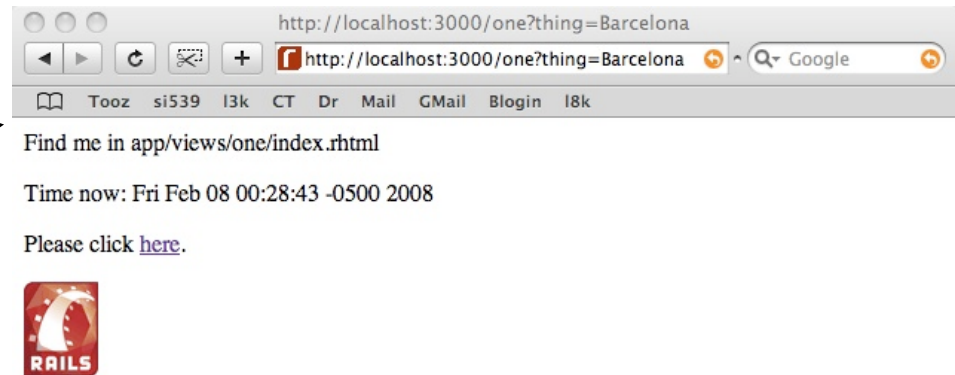
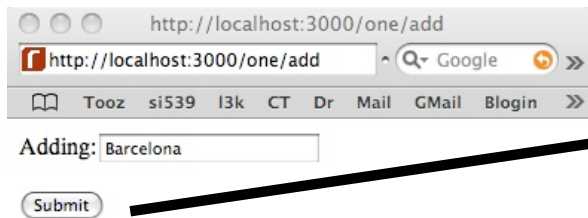
- Looking at Rails log output is important - it helps you figure out why something is not working.
- The logs look complex, messy, and confusing
- But after a while, if you look long enough - you can see through the log and visualize the activity in your program (as per Tank in the Matrix)



Where are the Logs?

- The log you are interested in is stored in `log\development.log`
- On Macintosh, this log scrolls by continuously in the terminal window while your server is running.
- On Windows - you use a program like WinTail to open the log file - WinTail watches the “tail end” of the file and immediately shows you if something new appears.





Processing OneController#index (for 127.0.0.1 at 2008-02-08 01:13:08) [GET]

Session ID: 7c1873d5af186e12f339014a1d5ab7d6

Parameters: {"action"=>"index", "controller"=>"one", "thing"=>"Barcelona"}

WE ARE IN THE INDEX ACTION!

Rendering one/index

Completed in 0.01026 (97 reqs/sec) | Rendering: 0.00751 (73%) | 200 OK

[http://localhost/one?thing=Barcelona]

```
Processing OneController#index (for 127.0.0.1 at 2008-02-08 01:13:08) [GET]  
  Session ID: 7c1873d5af186e12f339014a1d5ab7d6  
  Parameters: {"action"=>"index", "controller"=>"one", "thing"=>"Barcelona"}  
WE ARE IN THE INDEX ACTION!  
Rendering one/index  
Completed in 0.01026 (97 reqs/sec) | Rendering: 0.00751 (73%) | 200 OK  
[http://localhost/one?thing=Barcelona]
```

Rails receives a GET request from our browser.

The Session identifies which browser sent the request if multiple browsers are active

The parameters include our data (Barcelona) from the form field named “thing”.

You see the log message from within the INDEX action.

When the action finished it moves on to render the view file (index.rhtml) one/index

Then it is all finished - it took about 1/100 of a second to do the work


Using the Form Data

- We will change the action to look at the form data. If the user entered “Barcelona”, we will say “Hola” - otherwise we say “Hello”
- We pass the greeting from the action to the view in a variable named `@greet`
- Variables that start with `@` in Ruby are special

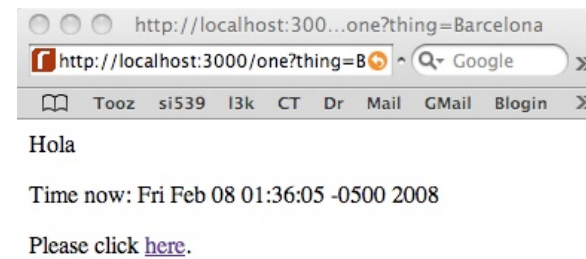
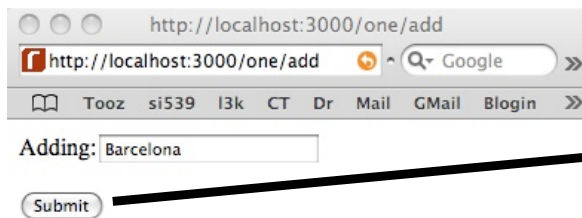
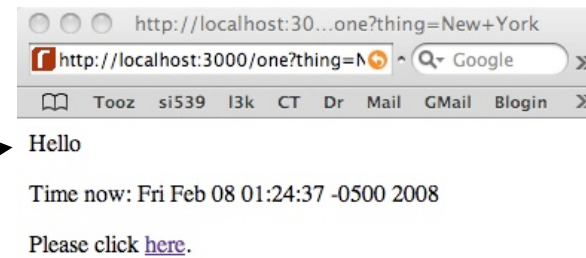
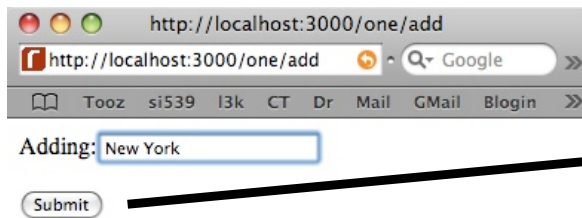
```
def index
  if params[:thing] == "Barcelona"
    @greet = "Hola"
  else
    @greet = "Hello"
  end
end
```

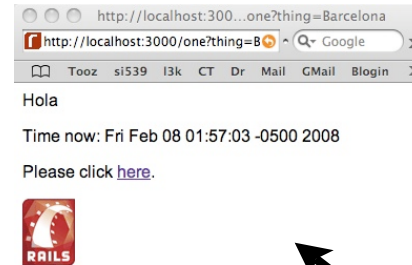
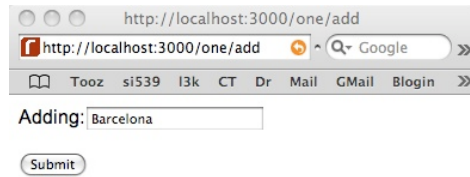
Displaying the Greeting

- We add some more Embedded Ruby to the view file index.rhtml
- The code prints out the contents of the `@greet` variable



```
<p><%= @greet %></p>
<p>Time now: <%= Time.now %></p>
<p>Please click
<a href="<%= url_for :action => "add" %>" >
here</a>.
<p>
<%= image_tag "rails.png", :alt => "Logo" %>
</p>
```



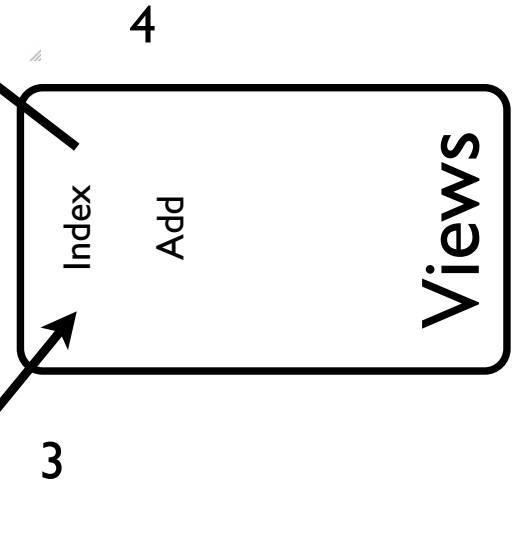


GET /one/index?thing=Barcelona

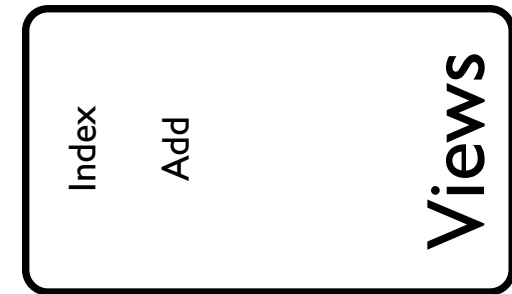
Our index action now looks at data from a form, and makes a decision based on that data. The index action then produces data (@greet) which is passed into the index.rhtml view for display to the user.

Index
Add
@greet

One




```
<p><%= @greet %></p>
<p>Time now: <%= Time.now %></p>
<p>Please click
<a href="<%= url_for :action => "add" %>" >here</a>.
<p><%= image_tag "rails.png", :alt => "Rails Logo" %></p>
```



```
def index
  if params[:thing] == "Barcelona"
    @greet = "Hola"
  else
    @greet = "Hello"
  end
end
```

Index
Add

One

The Controller / Actions are written in Ruby - the Views are written in HTML with Embedded Ruby.

Summary

- A Rails application has a particular directory structure and logical flow
- The Rails approach to MVC is embedded in its directory structure
- Initially it might be hard to find your way around because the structure is provided for you
- Later - having identical program structure makes it really simple to look at Rails applications developed but others